INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig
Maximilians
Universität
München
LMU

Problem Solving with Model-Generation Approaches based on PUHR Tableaux

Norbert Eisinger, Tim Geisler

In Proc. *Problem-solving Methodologies with Automated Deduction*, workshop at CADE '98
http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-1998-8, May 1998

Problem Solving with Model-Generation Approaches based on PUHR Tableaux

Norbert Eisinger and Tim Geisler

Institut für Informatik, Universität München, Oettingenstr. 67, D-80538 München
{eisinger|geisler}@informatik.uni-muenchen.de

Abstract. The workshop on "Problem-solving Methodologies with Automated Deduction" at CADE-15 is based on a collection of real-world problems from six practical domains. This paper describes and analyses the problem-solving process for these problems using the PUHR tableau method and some of its refinements.
The paper presents some general criteria to decide whether a problem domain may be amenable to the method. For instance, problems in the domain should be representable without a full treatment of equality. Three out of the six given domains meet these criteria. We could solve the problems from these domains with only minor modifications of the problem representation, with neither fine-tuning of system parameters nor run-time user interaction, and with satisfactory results.

1 Model Generation with PUHR Tableaux Approaches

Model generation was introduced in the late 80s as a new paradigm for automated deduction [MB88], which was motivated by database applications. The suggestion was to shift the focus prevailing at the time from demonstrating the *unsatisfiability* of first-order formulas to demonstrating their *satisfiability* by constructing satisfying Herbrand interpretations, i. e., models.

The approach presented in [MB88] combines a special form of hyper-resolution with a beta or splitting rule, thus representing an early attempt at reconciliation between the resolution and the tableau framework. A formalisation called PUHR tableaux [BY96] makes this hybridity quite explicit. Like traditional tableau systems, the PUHR tableau method enjoys not only the usual refutation completeness of resolution systems, but also a number of properties with respect to satisfiability. Like traditional resolution systems, the PUHR tableau method avoids the usual problems with the gamma rule of tableau systems by using unification to integrate instantiation into the modus ponens rule.

The PUHR tableau method expects the input formulas to be represented as a set of clauses in implication form. Application of the *positive unit hyper-resolution* rule amounts to a forward reasoning process, which in the case of Horn clauses corresponds to the standard fixpoint iteration. Disjunctions resulting from non-Horn clauses are handled by the *splitting* rule, which introduces branching into the forward reasoning process. Normally such a splitting might be critical if atoms in a disjunction share variables. However, a special syntactic

requirement on the input clauses, *range restriction*, ensures that disjunctions always become ground instantiated and thus uncritical before splitting. As a by-product, unification for the positive unit hyper-resolution rule actually simplifies to matching.

Range restriction means that every variable of a clause occurs in a negative literal. Many practical problem domains naturally meet this requirement, but some don't: a reflexivity axiom ranging over the entire universe of discourse is a typical counterexample. In such cases the clauses can always be transformed into a range-restricted form, hence the requirement does not impose a fundamental limitation. However, in certain cases the transformation may result in a problem representation that is hard to handle efficiently.

Following a common and well-tried database tradition, PUHR tableaux represent only the positive part of interpretations, leaving negative information implicit. This reduces the search space and allows a direct Prolog implementation mapping the tableau structure to the Prolog search tree and representing derived atoms in the Prolog database.

The SATCHMO programs in [MB88] are remarkably short and simple, yet efficient. They provide the basis for a wide range of variants implementing alternative search strategies, fairness conditions, data representations, etc. [BM95]. Other variants are concerned with efficiency enhancing techniques, in particular incrementality and compilation [SG96].

In order to meet the needs of applications, some refinements have been developed that increase the power of the underlying PUHR tableau method. One of them, *complement splitting*—called *folding down* in [LMG94]—was already introduced in [MB88]. Complement splitting contributes not only to efficiency, but also to a method that generates all minimal models but no non-minimal ones [BY96]. Another, called *extended delta rule*, contributes to the EP tableau method [BT97], which is complete for both unsatisfiability and finite satisfiability. The latter refinement handles non-clausal formulas and treats existential quantifiers in a way different from Skolemization. It has been applied to the constraint satisfiability problem during the schema design phase of databases [BEST98].

In order to tackle the given workshop problems, we used various refinements of model generation approaches based on PUHR tableaux.

2 Systems

We have access to a number of implementations focussing on different refinements of the PUHR tableau method. In order to avoid completely heterogeneous results that would be hard to compare, we chose a single system as a uniform basis for this paper: FUNCTIONAL SATCHMO [GPS97]. This implementation happens to include all refinements we wanted to consider. It is written in the lazy functional programming language Haskell.

Following the structure that will also be used in subsequent sections about individual problem domains, the characteristics of FUNCTIONAL SATCHMO can be described as follows.

Problem Formulation. FUNCTIONAL SATCHMO accepts a slightly modified OTTER syntax for clauses and for first-order formulas. A special built-in predicate $\neq$ for syntactic inequality of ground terms is available.

FUNCTIONAL SATCHMO offers automatic transformations of first-order formulas into clauses and of clauses into range-restricted clauses. It can also transform into a clause-like form preserving existential quantifiers, i. e., without Skolemization. This transformation may introduce auxiliary predicate symbols.

Inference Methods. FUNCTIONAL SATCHMO implements the PUHR tableau method [BY96]. It is incremental and orders competing hyper-resolution steps by increasing size of the hyper-resolvents. This is not in general a fair strategy, but fairness is not needed for the problems given in the workshop web page.

As an alternative to the standard rule for splitting disjunctions of atoms, a rule for complement splitting is offered:

$$\frac{A_1 \vee A_2 \vee \ldots \vee A_n}{\begin{array}{c|c|c|c} A_1 & & & \\ \neg A_2 & A_2 & & \\ \vdots & \vdots & & \\ \neg A_n & \neg A_n & \ldots & A_n \end{array}}$$

Existential quantifiers can optionally be handled by the extended delta rule:

$$\frac{\exists x\, \varphi[x]}{\varphi[c_1] \mid \ldots \mid \varphi[c_n] \mid \varphi[c_{new}]}$$

$c_1, \ldots, c_n$ are the constants in the branch,
c_{new} is a constant distinct from them.

Together with the transformation that preserves existential quantifiers, this rule corresponds to a restricted form of the EP tableau method.

User Interaction. Neither required nor offered.

Tuning of System Parameters. The only tuning provided is the optional use of the complement splitting rule. However, there is not much to tune. In our experience it is almost always useful to use it.

Presentation of the Solution. The solution is a sequence of Herbrand models, each represented as the set of ground atoms it satisfies. The sequence is empty iff the input set of formulas is unsatisfiable.

For problems with a small to medium-size search space, the tableau browser SNARKS [KE97] offers a graphical interface to explore a tableau. SNARKS is tightly integrated with Prolog implementations of the PUHR tableau method and of the EP tableau method, but only loosely coupled with FUNCTIONAL SATCHMO.

3 Selection of Suitable Problem Domains

Like any deduction method, the PUHR tableau method is not equally well-suited for all problem classes alike. Therefore we list some criteria, gained from experience, to decide whether a given problem domain is likely to be amenable to the approach. We then apply the criteria to the six problem domains given for the workshop.

3.1 Selection Criteria

The PUHR tableau method is appropriate for problem domains where the underlying deduction task is to decide whether a set of first-order formulas is satisfiable or unsatisfiable or to decide whether it is finitely satisfiable – notwithstanding the undecidability of these questions in general.

The solutions should not be requested to consist in compact representations of proofs in the case of unsatisfiability. Instead, solutions should consist of representations of (finite) models, possibly minimal or otherwise distinguished ones.

It is advantageous if the "natural" representation for problems of the domain results in range-restricted clauses. Problems ought to be representable without presupposing built-in equality or other theories. Nevertheless, restricted forms of equality are possible if the unique-name assumption [GMN84] applies to the domain.

The PUHR tableau method is definitely not the method of choice for domains that require in-depth equality reasoning or infinite models.

3.2 Assessment of the Given Problem Domains

The *Mathematics and Logic* domain consists of problems that can be expected to be unsatisfiable and heavily depend on equality reasoning. Model-generation techniques would seem out of place here. The *Software Verification* and the *Reactive Systems* domains also require built-in equality and apparently other theories as well. Therefore we ruled out these domains.

The problems from the *Diagnosis* domain are satisfiable. Here the deduction task is to compute models that meet a certain minimality condition. The fact that most of the problems admit a propositional specification indicates that infinite models are ruled out. Model-generation techniques are promising for this domain and have in fact been applied in previous work [BFFN97].

Much of the same can be said about the *Planning* domain. The PUHR tableau method should be tested on this domain.

Finally, the *Natural Language Understanding* domain presents fairly small problems for which the satisfiability or unsatisfiability have to be decided. The precise deduction task is hard to understand from the material provided, it may or may not be the case that model minimality plays a role. The domains are finite, and although equality does occur, it seems to be restricted and the unique name assumption does apply at least in some cases. In some problems

the point seems to be that after Skolemization only infinite models exist – this is exactly the point addressed by the EP tableau method.

In the sequel we discuss those three domains in more detail.

4 Diagnosis

We briefly recall the formalisation of the diagnosis examples for electrical circuits. For each circuit, a set of clauses is given that models

- for each of the circuit's components its normal behaviour, under the condition that it is not abnormal and
- an observation that contradicts the behaviour of the correctly functioning circuit.

The clause sets are satisfiable. The deduction task is to compute all models of the given clause sets where the extension of the predicate describing the abnormality of a component ($ab/1$) is minimal. In particular, the assumption that there is at most one abnormal component is called the single-fault assumption. The name of the abnormal components can be read off from the models.

4.1 Solution Process

Problem Formulation. For each diagnosis problem, a normal and a renamed variant was given in the workshop web page. Furthermore, there is also a choice in the coding of the single-fault assumption.

Usage of the Renamed Variants. We have chosen the renamed variants of the problems, because it was shown that they provide an optimization for bottom-up tableau calculi in diagnosis applications [BFFN97]. In the same way the renamings can be expected to be advantageous for the PUHR tableau method.

Coding of the Single-Fault Assumption. The formulation of the single-fault assumption requires some design decisions.

At the clause level, the single-fault assumption can be axiomatized by enumerating all ground clauses that exclude models with at least two faults. This results in a huge number of clauses, which cannot be handled efficiently by FUNCTIONAL SATCHMO. Using predicate logic with the unique-name assumption and a built-in predicate $\neq$ for syntactic inequality, the single-fault assumption is ensured by a *single* axiom:

$$ab(X) \wedge ab(Y) \wedge X \neq Y \rightarrow \bot$$

An atom $X \neq Y$ is satisfied, iff X and Y are bound to different ground terms. It is an error if one of its arguments is not ground. Note that $\neq$ is a very general built-in predicate and not problem-specific. Of course, this approach can be generalized to the n-fault assumption.

At the implementation level, the single-fault assumption could be hardcoded into the deduction system. This would shift a part of the declarative specification of the deduction problem into a problem-dependent modification of the prover, thus restricting its generality.

We decided to use the single-axiom representation of the single-fault assumption. The necessary built-in $\neq$ is provided by FUNCTIONAL SATCHMO, anyway.

Inference Methods. We used the PUHR tableau method with complement splitting, but this had only very little influence on the times needed to compute the diagnoses – in contrast to results in the planning domain.

No Usage of the Lemma Technique. The given problem is to efficiently compute all minimal diagnoses. Baumgartner et al. [BFFN97] propose a lemma technique, which is used to ensure that no minimal diagnosis is computed twice: If a model containing $ab(c)$ is found, the new clause $\neg ab(c)$ is introduced in the search for subsequent models.

FUNCTIONAL SATCHMO would have to be changed slightly to incorporate the lemma technique. We wanted to assess the possible gains of such an application-dependent modification before deciding about changes to our system. First, we ran FUNCTIONAL SATCHMO on the given problem specifications in order to generate all minimal diagnoses. For the given problems, every diagnosis was computed only once. Thus the lemma technique would not have any effect on the result. Second, we augmented the given problem specifications with the negations of all minimal diagnoses. This turned the model-generation task into a refutation task that could serve as an estimate for the pruning effects of the lemma technique. If these effects were high, the runtimes for the augmented problem specifications would be considerably lower than for the original problem specifications. However, the differences in runtime were negligible. Thus the lemma technique would not have much pruning effect for the given problems. Therefore we did not bother to integrate the lemma technique into our system.

4.2 Results

The results are summarized in Table 1. We give the times[1] to compute all minimal single-fault diagnoses, and the times to compute all diagnoses, but stopping as soon as all diagnoses were computed (in this case, the number of diagnoses was given to the system).

Note that for problem c499, the two diagnoses are computed very fast, whereas it takes a long time to prove that these two diagnoses are the only ones. In contrast, for problem c880, these times are approximately equal.

[1] The times needed to read and parse the specifications are excluded (which are less than 5 seconds). All runtimes presented in this paper are taken on a Pentium Pro with 180 MHz running Linux. FUNCTIONAL SATCHMO is compiled with the Glasgow Haskell Compiler version 3.01 using optimization.

A possible explanation for this phenomenon is in the structure of the search space. The *ab* literals occur only as the leftmost positive literals in clauses. Therefore, the models representing minimal diagnoses tend to occur on the left-hand side of the search tree. If there are just a few models, it can be expected that they are computed early. If there are a lot of models, some of them are likely to occur on the right-hand side of the search tree and are thus computed near the end of the model search.

Table 1. Runtimes for the diagnosis problems. "all" means time needed to compute all diagnoses. "first n" means time to compute all diagnoses, but stopping after the given number n of diagnoses (i. e., 2 or 19) were computed.

Problem	Number n of Diagnoses	all	first n
c499	2	49.7s	0.3s
c880	19	5.9s	5.5s

4.3 Questions

Subsumption. Some of the clauses for the diagnosis examples are redundant, because they can be subsumed by other clauses. The elimination of subsumed clauses leads to reductions of the problem size of up to 10% for the given examples.

Furthermore, a preprocessing with unit resolution should further reduce the problem size and thus the search space.

Nonground Representation. The clauses resulting from the description of the behaviour of components of the circuit are ground. For each component of the same class, there are formulas describing the behaviour of each instance. E. g., for the two inverters in the example inv-inv-or we have:

$$\neg ab(inv1) \rightarrow (high(inv1, o) \leftrightarrow \neg high(inv1, i))$$
$$\neg ab(inv2) \rightarrow (high(inv2, o) \leftrightarrow \neg high(inv2, i))$$

It would be worth while to evaluate a first-order formulation:

$$inverter(inv1).$$
$$inverter(inv2).$$
$$\forall I\ inverter(I) \wedge \neg ab(I) \rightarrow (high(I, o) \leftrightarrow \neg high(I, i))$$

Unfortunately, the tools used to convert the given ISCAS benchmark problems to the clausal representation were not supplied. We did not try to reintroduce variables to the given ground problems.

5 Planning

We briefly recall the formalisation of the planning examples from the blocks world. For each planning problem, a set of first-order formulas is given that models

- the domain axioms
- the initial state and the goal state
- the actions that define state transitions

All predicates describing time-varying conditions and actions have a time argument. Time is discrete and consists of a fixed number of time instances. Existential quantifications are expressed by finite disjunctions.

The formulas are satisfiable. The deduction task is to compute all models of the given formulas. A plan can be read off from a model of the formulas. Minimality of models in the set-theoretic sense does not seem to be relevant.

5.1 Solution Process

Problem Formulation. For each planning problem, both a propositional logic formulation and a first-order formulation was given.

Usage of First-Order Formulation. We have chosen to apply FUNCTIONAL SATCHMO only to the first-order formulation of the planning problems because it is a first-order prover and not optimized for propositional problems. The first-order formulation was not clausified in advance.

The Predicate neq. In the given first-order formalizations of the problems, the predicate *neq* is defined by means of an auxiliary predicate *different*:

$$\forall X, Y\ \mathit{different}(X, Y) \vee \mathit{different}(Y, X) \rightarrow \mathit{neq}(X, Y).$$

The predicate *different* is defined by facts for all appropriate constants occurring in the specification. We replaced *neq* optionally with the built-in syntactic inequality $\neq$, relying on the unique name assumption.

Inference Methods. We applied the PUHR tableau method with and without complement splitting – with different runtimes.

5.2 Results

We ran FUNCTIONAL SATCHMO with optional complement splitting both on the unmodified problem formulation and on a variant, where all occurrences of *neq* are replaced by the built-in syntactic inequality $\neq$. Table 2 gives the times needed to find all minimal models, and the times to find the first model of the respective problem.

Table 2. Runtime results for the first-order planning problems. Times in seconds. "cs" means "complement splitting".

Problem	#Models	All Models				First Model			
		with *neq*		with $\neq$		with *neq*		with $\neq$	
		no cs	with cs	no cs	with cs	no cs	with cs	no cs	with cs
tiny	1	0.3s	0.3s	0.2s	0.2s	0.2s	0.2s	0.2s	0.2s
anomaly	1	1.3s	1.0s	1.4s	0.9s	0.6s	0.6s	1.1s	0.6s
reversal	1	1.7s	1.5s	2.1s	2.1s	1.2s	1.1s	1.6s	1.4s
medium	16	26.0s	12.8s	37.9s	14.0s	9.0s	5.3s	11.3s	6.2s
huge	9	2932.5s	2310.4s	220.2s	200.1s	251.5s	212.2s	175.3s	162.1s

Compared to the times given in the workshop web page (the stochastic prover Walksat needed only 0.6s with the huge problem), the runtimes with FUNCTIONAL SATCHMO are not very good. Nevertheless, they are much less than the times given for the systematic, hyper tableau based prover Nihil (which needed 146742s for solving the huge problem). We believe, however, that a stochastic prover is only of limited use. Since different models correspond to different plans, it is reasonable to assume that the application really requires the generation of all models rather than just an arbitrary one. This is not possible with a stochastic prover but is the normal mode of using the PUHR tableau method.

Complement splitting generally improves the runtimes, especially for the larger problems.

The usage of the built-in syntactic inequality instead of the explicit axiomatization of *neq* results in larger runtimes for the smaller examples, but reduces the runtimes for the huge example. It is quite remarkable that with the built-in syntactic inequality the computation of all minimal models is only slightly more expensive than the computation of the first model – in contrast to the ten-fold increase in runtime with the explicit axiomatization of *neq*.

A reason for these phenomena could be the difference of the search spaces for the respective problems in combination with incremental evaluation. The derivation of (positive) *neq* literals can be used to instantiate clauses with occurrences of negative *neq* literals and thus may initiate hyper-resolution steps with these clauses. In contrast, negative $\neq$ literals act just as filters guarding the applicability of hyper-resolution steps.

5.3 Questions

Usage of Linear Encodings? Kautz and Selman [KS92] propose the usage of linear encodings of certain predicates[2] in order to reduce the number of propositional variables in the propositional logic instantiation of the first-order planning problem. In contrast, a first-order prover instantiates these formulas as needed. It

[2] For example, instead of a predicate $move(X, Y, Z, I)$ three predicates $object(X, I)$, $source(Y, I)$, and $destination(Z, I)$ are used.

would be interesting to evaluate the differences of using linear encodings instead of the original encodings for first-order provers.

6 Natural Language Understanding

The material provided for this domain was not fully comprehensive to us. It does not describe the deduction task in detail. In order to identify the deduction tasks, we tried "reverse engineering" from the formalisations in OTTER syntax. These formalisations are not very human-readable, because everything is encoded into a single, complex formula, and some of the problem specifications were even buggy. Moreover, our own background in computer linguistics is limited. With all due reservation, we characterise the domain as follows.

The meaning of a sentence can be represented by a *discourse representation structure*, which can be translated into a first-order formula (see [KR93]). Starting from some general *background knowledge*, the meaning of sentences in a discourse is successively accumulated. Let α denote the formulas representing this accumulated knowledge at a certain point, and assume that at that point a new sentence is uttered, which has the discourse representation structure D corresponding to a first-order formula $\varphi(D)$.

Now linguists are interested in three possible properties of the new sentence:

(i) *Global informativity:* The new sentence is informative iff it conveys information not entailed by the accumulated knowledge, i. e., iff $\alpha \models \varphi(D)$ does not hold, i. e., iff $\alpha \cup \{\neg\varphi(D)\}$ is satisfiable.
The deduction task then is to decide the latter. In the positive case each (minimal?) model represents an example state where the addition of the new sentence adds new information. In the negative case a representation of the refutation represents an explanation why the new sentence is not informative. It is not clear to us whether minimality of models is relevant, whether models and refutations are relevant, or whether a simple yes/no answer is all the application requires.

(ii) *Consistency:* The new sentence is consistent with the accumulated knowledge iff $\alpha \cup \{\varphi(D)\}$ is satisfiable.
Thus the deduction task is of the same nature as above. In the positive case each (minimal?) model represents an example state resulting from the incorporation of the new sentence. Again, we are not sure which kinds of answers are needed.

(iii) *Local Informativity:* Apparently there may be several possible meanings of the new sentence. It seems to us that different translations $\varphi_i(D)$ have to be considered, but that the deduction task is exactly the same as for global informativity.

Summarized, the primary deduction task seems to be to decide the satisfiability of first-order formulas. This is not possible in general, but it may be under reasonable additional assumptions, e. g. about the number of objects referred

to in the accumulated knowledge. Model generators can provide yes/no answers and also the models that might be the basis of explanations[3].

6.1 Solution Process

Problem Formulation. The problems are formalized as first-order formulas. Some of them contain equality literals and/or existential quantifiers.

Equality and Skolemization. In some of the given problems (problem sets 3 to 5) equality is used in the problem formulation. It is possible to augment the examples with an explicit axiomatization of equality, consisting of the usual reflexivity, transitivity, symmetry, and substitution axioms.

For the PUHR tableau method, the reflexivity and substitution axioms may be problematic if there are function symbols. The reflexivity axiom has to be transformed into range-restricted form, which in turn requires axioms enumerating the Herbrand universe. These axioms and the substitution axioms lead to a generation of increasingly nested terms if function symbols are present.

However, the given problems do not contain function symbols.

Instead, they contain existentially quantified variables, such that Skolemization might introduce function symbols. Skolemization may transform a finitely satisfiable set of formulas into a form with an infinite Herbrand universe where no finite subset of the Herbrand base represents a model.

This effect can be avoided by using the extended delta rule for existential quantifiers [BT97].

Equality Elimination. In the given examples, the equality predicate is used in two different syntactic situations:

- one argument is ground (e. g., $X = c$)
- both arguments are variables

Looking at the former situation in detail, the occurrences of the variable X show one of the patterns

$$\forall X\ X = c \rightarrow \psi[X] \quad \text{or} \quad \exists X\ X = c \wedge \psi[X].$$

The two patterns can be transformed into $\psi[c]$.

Note that the problems for Problem Set 2 are exactly the result from applying this transformation to the problems for Problem Set 3.

[3] For the purposes of this workshop, model-generation approaches were useful in yet another way: some of the models pointed out some bugs in the problem specifications that could then be corrected.

Inference Methods. In problems with existentially quantified variables, we have a choice between

- Skolemizing the existentially quantified variables and
- using the extended delta rule.

The two methods may differ in their termination behaviour and in the set of models they generated. We applied both methods to all of the problems.

6.2 Results

All given problems are very easy with very small search spaces, if they can be handled by the respective method at all. The runtimes are in the range of some milliseconds and are therefore not given here.

Problem Set 1 ***"Informativity".*** For all problems the satisfiability status can be determined both by the PUHR tableau method and the EP tableau method. One problem, d2r1i, is unsatisfiable, all others are satisfiable.

The discourse of this problem set is given by two sentences:

"Mia has a husband. She is married."

Let D_1 be the discourse representation structure of the first sentence and D_2 of the second. Let $\varphi(D_i)$ be the corresponding first-order formulas. $\varphi(D_1)$ contains an existentially quantified (sub)formula: $\exists C\ husband(C) \wedge have(mia, C)$. Let β be the background knowledge.

Problem d1r1i is to test the informativity of the first sentence, i.e., to decide whether $\beta \cup \{\neg\varphi(D_1)\}$ is satisfiable. FUNCTIONAL SATCHMO generates a single minimal model: $\{woman(mia)\}$, thus the first sentence is informative.

Problem d2r1i is to test the informativity of the second sentence, i.e., to decide whether $\beta \cup \{\varphi(D_1)\} \cup \{\neg\varphi(D_2)\}$ is satisfiable. FUNCTIONAL SATCHMO shows the unsatisfiability, thus the second sentence is not informative and the formula $married(mia)$ need not be added to the accumulated knowledge.

Somewhat contrary to the label "informativity" given to the entire problem set, the remaining problems test the consistency of the sentences above. Here the results depend on whether existential quantifiers are treated by Skolemization or by the extended delta rule.

Problem d1r1ii is to decide whether $\beta \cup \{\varphi(D_1)\}$ is satisfiable. There is one minimal model of the Skolemized formula:

$$\{woman(mia),\ married(mia),\ have(mia, c_1),\ of(c_1, mia),\ husband(c_1)\}$$

Under the (implicit) assumption that mia and c_1 are distinct, this model looks very reasonable. Using the extended delta rule, there are two minimal models:

1. $\{woman(mia),\ married(mia),\ have(mia, mia),\ of(mia, mia),\ husband(mia)\}$
2. $\{woman(mia),\ married(mia),\ have(mia, c_1),\ of(c_1, mia),\ husband(c_1)\}$

In the first model Mia is married to herself. The fact that this is a model may be an indication that the background knowledge β is specified inappropriately or incompletely in the given formulation. The second model corresponds to the one found for the Skolemized form.

Problem d2r1ii is to decide whether $\beta \cup \{\varphi(D_1)\} \cup \{\varphi(D_2)\}$ is satisfiable. There is one minimal model of the Skolemized formula:

$$\{woman(mia),\ married(mia),\ have(mia, c_1),\ have(mia, c_2),\\ of(c_1, mia),\ of(c_2, mia),\ husband(c_1),\ husband(c_2)\}$$

Based on the (implicit) assumption that *mia* and c_1 and c_2 are pairwise distinct, this means that Mia is married with two husbands! The reason for this unexpected result may again be a deficiency in the given formalisation of the background knowledge β, but more likely it is a consequence of Skolemization. The Skolemized form is satisfiable iff the original formula is, but the set of models of the two formulas are not in general the same.

Using the extended delta rule, we get the same minimal models as for d1r1ii. Assuming that the first of them is caused by a bug in the background knowledge β, the extended delta rule leads to more natural models than Skolemization.

The subsequent descriptions for the other problem sets do not discuss the generated models in detail.

Problem Set 2 ***"Presupposition Projection".*** For all problems the satisfiability status can be determined by FUNCTIONAL SATCHMO using either Skolemization or the extended delta rule. All problems are satisfiable, except for problem d2r1iii1, which is unsatisfiable.

Problem Set 3 ***"Presupposition Projection".*** The problems of this set are formulated with many equations of the form $X = c$. Eliminating these equations in a preprocessing step as described above, we obtain exactly the problems of the previous set and thus the same results.

Problem Set 4 ***"One Boxer".*** The discourse of this problem set is given by four sentences:

> "Mia loves Butch and Vincent. Butch is a boxer. Vincent is a boxer.
> Mia loves one boxer."

Note that the last sentence refers to the *number* of boxers loved by Mia.

FUNCTIONAL SATCHMO with Skolemization determines the satisfiability status for all problems except d4r1i, for which it does not terminate. With the extended delta rule, the satisfiability status can be determined for all problems. All problems are satisfiable.

The important problem, d4r1i, is to test the informativity of the last sentence, i.e., to decide whether $\beta \cup \{\varphi(D_1), \varphi(D_2), \varphi(D_3)\} \cup \{\neg\varphi(D_4)\}$ is satisfiable.

The formalisation contains a formula that is problematic for calculi requiring Skolemization:

$$\forall D\ boxer(D) \wedge love(mia, D) \rightarrow (\exists E\ boxer(E) \wedge love(mia, E) \wedge \neg(D = E))$$

Skolemizing E with e leads to:

$$\forall D\ boxer(D) \wedge love(mia, D) \rightarrow (boxer(e(D)) \wedge love(mia, e(D)) \wedge \neg(D = e(D)))$$

Otter and FUNCTIONAL SATCHMO with Skolemization do not terminate here: they keep generating boxers named with terms constructed by increasingly nesting the Skolem function symbol e.

By using the extended delta rule instead of Skolemizing, the problem can be solved. FUNCTIONAL SATCHMO generates in this case the minimal model:

$$\begin{aligned}&\{love(mia, butch),\ love(mia, vincent),\\ &\ vincent = vincent,\ mia = mia,\ butch = butch,\\ &\ boxer(butch),\ boxer(vincent)\}\end{aligned}$$

Problem Set 5 ***"One Boxer (II)".*** FUNCTIONAL SATCHMO with Skolemization determines the satisfiability status for all problems except d4r1i, for which it does not terminate. With the extended delta rule, the satisfiability status can be determined for all problems. All problems are satisfiable, except problem d4r1ii, which is unsatisfiable.

7 Conclusions

We have presented some general criteria to decide whether a problem domain may be amenable to model-generation approaches based on the PUHR tableau method.

We assessed the problem domains provided for the workshop with these criteria and identified three domains to be tested with the PUHR tableau method: diagnosis, planning, and natural language understanding. The other domains could be quickly ruled out. We obtained satisfactory results for each of these domains without modifying the given problem specifications or adapting the system to the problems.

The fact that the clausal transformations of the formulas in the diagnosis domain and the planning domain contain only range-restricted clauses corroborates the claim that range-restriction is natural for many applications. Note that for these clausal transformations the improvements of the hyper tableaux methods [BFN96,Bau98] have no effect, because these methods coincide with the PUHR tableaux method for range-restricted clauses.

The built-in syntactic inequality $\neq$ is useful for short and maintainable specifications as well as for efficiency. Complement splitting contributed to efficiency as well.

As remarked in the planning domain, the usual set-theoretic notion of minimal models (sometimes restricted to a sublanguage as in the diagnosis domain) is not always appropriate.

The notion of finite satisfiability seems relevant for the problems from the natural language understanding domain in two aspects. First, the techniques developed for finite satisfiability (the extended delta rule) make it possible to decide the given problems. Second and most important, we believe that if models are of interest in the application, then the models derivable by the extended delta rule are more natural than those derivable after Skolemization. Skolemization extends the signature and introduces complex names for objects that have only simple names in the un-Skolemized form. Combined with the unique name assumption this may distort the intended meaning of the formulas. In such cases approaches that do not need Skolemization would seem to be more appropriate.

Acknowledgements

We thank François Bry and Sunna Torge for helpful comments and discussions, and Peter Baumgartner and Johan Bos for their help with the workshop examples. Sven Panne and Heribert Schütz participated in the implementation of FUNCTIONAL SATCHMO and Martin Josko installed parts of the required software.

References

[Bau98] Peter Baumgartner. Hyper tableaux — the next generation. In Harrie de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1397 of *Springer LNAI*, 1998.

[BEST98] François Bry, Norbert Eisinger, Heribert Schütz, and Sunna Torge. SIC: Satisfiability checking for integrity constraints. In Piero Fraternali, Ulrich Geske, Carolina Ruiz, and Dietmar Seipel, editors, *Proc. 6th International Workshop on Deductive Databases and Logic Programming, DDLP '98*, GMD Report 22, Manchester, UK, June 1998.

[BFFN97] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Tableaux for diagnosis applications. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 97*, number 1071 in Springer LNCS, pages 76–90, Pont-a-Mousson, France, May 1997. Springer-Verlag.

[BFN96] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper tableaux. In *Logics in Artificial Intelligence: European Workshop, JELIA '96*, number 1126 in Springer LNAI, 1996.

[BM95] François Bry and Rainer Manthey. Variationen über ein Thema: Suchstrategien und Datenstrukturen für SATCHMO-Beweiser. In Andreas Krall and Ulrich Geske, editors, *Proc. 11. Workshop Logische Programmierung*, GMD-Studien Nr. 270, pages 205–216, 1995. In German.

[BT97] François Bry and Sunna Torge. Model generation for applications – a tableaux method complete for finite satisfiability. Technical Report PMS-FB-1997-15, Institut für Informatik, LMU München, 1997.

[BY96] François Bry and Adnan Yahya. Minimal model generation with positive unit hyper-resolution tableaux. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings, 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, number 1071 in Springer LNCS, pages 143–159, Terrasini, Palermo, Italy, May 1996. Springer-Verlag.

[GMN84] Hervé Gallaire, Jack Minker, and Jean-Marie Nicolas Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, 1984.

[GPS97] Tim Geisler, Sven Panne, and Heribert Schütz. Satchmo: The compiling and functional variants. *Journal of Automated Reasoning*, 18(2):227–236, 1997.

[KE97] Mathias Kettner and Norbert Eisinger. The tableau browser SNARKS (system description). In William McCune, editor, *14th Int. Conf. on Automated Deduction (CADE)*, volume 1249, pages 408–411. Springer LNAI, 1997.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Kluwer, 1993.

[KS92] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. ECAI '92*, Vienna, 1992.

[LMG94] Reinhold Letz, Klaus Mayr, and Christoph Goller. Controlled integration of the cut rule into connection tableau calculi. *Journal of Automated Reasoning*, 13:297–337, 1994.

[MB88] Rainer Manthey and François Bry. SATCHMO: A theorem prover implemented in Prolog. In E. L. Lusk and R. A. Overbeek, editors, *9th Int. Conf. on Automated Deduction (CADE)*, number 310 in Springer LNCS, pages 415–434, Argonne, IL, USA, 1988. Springer-Verlag.

[SG96] Heribert Schütz and Tim Geisler. Efficient model generation through compilation. In Michael McRobbie and John Slaney, editors, *13th Int. Conf. on Automated Deduction (CADE)*, number 1104 in Springer LNAI, pages 433–447. Springer-Verlag, 1996.